Rotating range sensor approached for mobile robot obstacle detection and collision avoidance applications

Cite as: AIP Conference Proceedings **2333**, 130001 (2021); https://doi.org/10.1063/5.0041746
Published Online: 08 March 2021

Ahmed Ismaiel, and M. Yu. Filimonov

Rotating Range Sensor Approached for Mobile Robot Obstacle Detection and Collision Avoidance Applications

Ahmed Ismaiel[1, a)] and M. Yu. Filimonov[1, 2, b)]

[1)] *Ural Federal University, Yekaterinburg, Russia*
[2)] *Krasovskii Institute of Mathematics and Mechanics, Yekaterinburg, Russia*

[a)] Corresponding author: en.ismaiel@gmail.com
[b)] fmy@imm.uran.ru

Abstract. Range finder sensors are widely used in the obstacle detection and collision avoidance applications. In this research, we propose rotating range finder sensor that provides economic and efficient solution for mobile robot applications. Rotating approach is achieved by coupling the range sensors with servomotor. In this article, rotating approach model design, main parameters, equations and limitation are described. In addition, an algorithm is developed to control the rotation angle of the range sensor, extract data from the approach and analyze it. A case study of the rotating approach by implementing ultrasonic sensor is simulated and the results are obtained. Simulation platform Gazebo and ROS are used to simulate the rotating approach.

INTRODUCTION

Autonomous Mobile robots (AMR) are widely used in many applications like industrial, medical, space researches, shipping and many other applications, as they can conduct intelligent actions without human supervising. AMRs are begin used mainly to replace human labor in complicated, senseless, dirty or dangerous tasks like dealing with radioactive waste [1]. Monitoring and sensing the surrounding environment is a key function of AMRs as they have to make intelligent interactions with other entities. That's why sensors are the key component of any AMR system as they provide information either about the entire robot's framework as its speed or location, or about robot's environment like distance to other objects in the environment or light intensity. Rangefinder sensors are extensively used in Mobile Robots' applications to measure the distance between the robot's body and other static or dynamic objects in robot's environment. Two main types of rangefinder sensors are ultrasonic rangefinder (the one we use in this research) and laser rangefinder. Rangefinder sensors are using time-of-flight and propagation speed of electromagnetic waves (ultrasonic or light waves) to find the distance to other objects [2]. The main advantages of ultrasonic sensors over laser or infrared rangefinder sensors are that ultrasonic waves aren't affected by the color of objects, transparent surface or sunlight. However, the measurements of ultrasonic sensors can be affected if the object surface contains materials like foam that can absorb or distract ultrasonic waves [3]. Ultrasonic sensors are classified as exteroceptive and active sensors. They are exteroceptive because they provide the robot with information about the environment, and they are active because they emit energy into the environment and perceive the information gathered from environment reaction [4]. Ultrasonic sensors consist of a trigger transducer to transmit an ultrasonic waves and an echo transducer to receive the reflected ultrasonic waves. There are different types of ultrasonic sensors that may contain one transducer as trigger and echo simultaneously, or consist of two transducers, one to trigger and the other to echo the reflected wave. An ultrasonic wave is emitted by the trigger transducer and propagates in a normal direction to the sensor plane. Once the wave strikes any obstacle, it is reflected back and received by echo transducer. By knowing the speed of sound wave and time-to-flight we can get the distance to this object. The conventional approach to use rangefinder sensors in mobile robot applications is to fix the sensors in the front side of the mobile robot to find any obstacle that may appear in its path. The number of used range finder sensors is varies from one application to another depending on the total area needed to be covered by the sensors. One of the early applications of mobile robots equipped with ultrasonic range finder sensors for obstacle detection is Yamabico mobile robot [5]. Yamabico is equipped with ultrasonic sensor in its front side. Yamabico successfully detect objects exist within a narrow angle 25 degree in front of it. Yamabico was also used to improve programming tools to simulate sensor-based robots and to simulate the ultrasonic wave propagation for obstacle detection and distance calculation [6]. Other mobile robot applications used more than one range finder sensors to cover wider area. Quantity and distribution of the range finder sensors depends on the application of mobile robot and the designed behavior. For example, an application of a wheelchair for physically disabled people was equipped by eight ultrasonic range finder sensors. In this application the environment should be monitored from the four sides of the wheelchair, so that the sensors were distributed around the wheelchair as the following: three in the front, two for each side and one in the backside [7]. The more area needed to be covered by mobile robot the more range finder sensors will be used. For example, in [8] mobile robot is designed to follow the

Applications of Mathematics in Engineering and Economics (AMEE'20)
AIP Conf. Proc. 2333, 130001-1–130001-7; https://doi.org/10.1063/5.0041746
Published by AIP Publishing. 978-0-7354-4077-7/$30.00

walls in the environment, so that it is demanded to cover the area in the front, left, right sides of the mobile robot. In this application five ultrasonic sensors were fixed as the following: one sensor in the center front of the robot. two sensors in the left and right of the robot. One sensor between front and right sensors and last one between the front and left sensors. In [9] an autonomous navigation is investigated by a mobile robot equipped with sixteen ultrasonic sensors that distributed along the rectangular-shaped robot's frame distributed as four sensors in each corner, four sensors along each the right and left sides, two sensors along each of the front and back sides. In our research we are trying a new approach for using the rotating rangefinder sensors with autonomous robot. In this approach we use one rotating range sensor to cover a wider area instead of using multi sensors fixed in different positions on the robot body. This research is apart of model and design of differential drive mobile robot with navigation and collision avoidance algorithms [10].

ROTATING RANGE SENSOR APPROACH

Model Design

In this section, we explain the design of rotating range sensor approach and derive the mathematical model equations. Rotating approach model depends on a_{total} this is the total area need to be covered by the rotating approach and it is represented as a part of circle and described by angle of first chord arc θ_{start} and angle of last chord θ_{end} as shown in figure 1. Range of the sensor is represented by a cone and has the following parameters: distance range d_{range} and

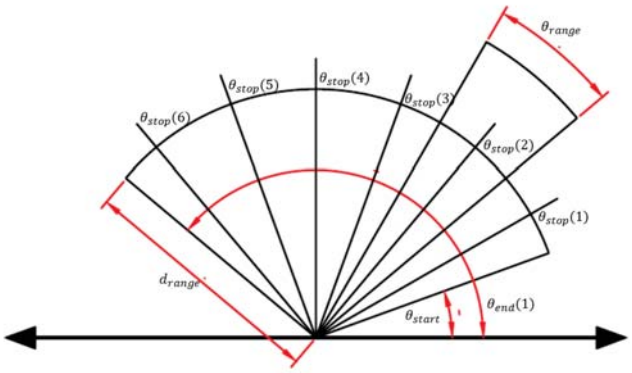


FIGURE 1. Rotating range sensor approach model.

angle range θ_{range} , distance range of the sensor d_{range} is the length of the cone, angle range θ_{range} are the cone angle. Range sensor is used to calculate the distance to any obstacle exists only inside the cone, for any obstacle located outside the cone, range sensor returns d_{range}. In the rotating range sensor approach, sensor rotates in two dimensions between θ_{start} and θ_{end} to cover the desired area a_{total}. The rotation behavior is not continuous in the way that sensor stops at specific angles θ_{stop}. It also rotates in both directions; clockwise and counterclockwise. Quantity of stop angles N and value of each stop angles are calculated by equation (1) and (2) respectively.

$$N = \frac{\theta_{end} - \theta_{start}}{\theta_{range}} \tag{1}$$

$$\theta_{stop}(n) = (\frac{2n-1}{2})\theta_{range} + \theta_{start} \qquad n \in \mathbb{Z} \cap [1, N] \tag{2}$$

Range sensor start position is $\theta_{stop}(1)$ and it holds there for a specific period t_{hold}, then rotates counterclockwise to $\theta_{stop}(2)$ and again holds for the same period t_{hold}. Sensor repeats this behavior for each θ_{stop}. When the sensor reaches $\theta_{stop}(N)$, it again repeats the previous behavior but in clockwise direction till $\theta_{stop}(1)$. Rotation angular velocity of

the sensor between stops is ω_m and it depends on the attached servomotor characteristics and mobile robot dynamic model. We get the moving time between two sequential stop angles $t_{move} = \frac{\theta_{range}}{\omega_m}$. From equation (3) we calculate the total time of one complete cycle t_{total}, this is the time that sensor takes to move from $\theta_{stop}(1)$ to $\theta_{stop}(N)$ or vise versa.

$$t_{total} = Nt_{stop} + (N-1)t_{move} \tag{3}$$

Equations (2)—(3) are representing the main design equations to get the configuration parameters of the rotating range sensor approach. The design sequence starts by determining a_{total}, θ_{start} and θ_{end} of the area desired to be covered by the rotating sensor approach. Then, calculate the stop angles $\theta_{stop}(n)$. Finally, program the controller algorithm to control the rotation of the range sensor in both directions clockwise and counterclockwise as mentioned before. The stop time θ_{stop} depends on the range sensor characteristics, this is the time needed by the range sensor to get the distance to the obstacle in its range.

Worst Scenario

In rotating range sensor approach, range sensor scans the area a_{total} by dividing it into portions. In each θ_{stop} the sensor scans a portion of a_{total} and by the end of each cycle, all the portions of a_{total} will be scanned by the sensor. As a result, any obstacle located inside a_{total} will spend a specific time before it is scanned by the rotating sensor, this period of time is called the blind time t_{blind} as the obstacles are located in the blind area that hasn't been scanned yet by the range sensor. The worst scenario represents the longest period that obstacle is located inside a_{total} before it is discovered by the sensor. This case happened when the range sensor at the beginning of the cycle $\theta_{stop}(1)$ or $\theta_{stop}(N)$ then it moves to the next θ_{stop} when an obstacle appears in the portion that has been just scanned by the sensor. That requires approximately two cycles in order to the range sensor scan the portion where the obstacle is located. The worst scenario and the maximum period of time any obstacle can spend before it been discovered by range sensor is $t_{blind\ max} = 2t_{total}$.

Rotating Range Sensor Limitations

In collision avoidance application for mobile robots, we define the term of robot critical time $t_{critical}$, this is the minimum time needed by the mobile robot to manipulate and avoid the collision with obstacles in its environment. $t_{critical}$ mainly depends on robot design and its control system. We assume that the worst scenario happens when the mobile robot moves with linear velocity v_r toward the obstacle. In this case, the distance between the obstacle and the robot will be $d_{blind\ max} = v_r t_{blind\ max}$. The limitation condition of the rotating range sensor approach is shown in equation (4). If $t_{critical}$ doesn't fulfil the limitation condition, the rotating approach in some cases may fail to give the data about the distance to the surrounding obstacles in suitable time.

$$t_{critical} > \frac{d_{range} - d_{blind\ max}}{v_r} \tag{4}$$

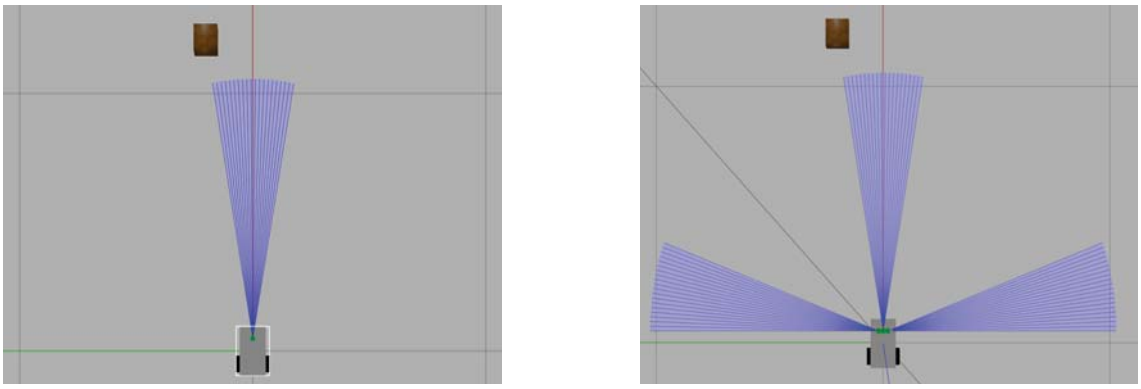
ROTATING ULTRASONIC SENSOR APPROACH

As mentioned in the Introduction that ultrasonic sensor is a member of rangefinder sensors family. It depends on emitting ultrasonic wave that strikes the obstacle, then calculates the distance to this obstacle by calculating the passed time between the emitted wave and the reflected one by the obstacles. In this section, rotating ultrasonic sensor approach is designed and simulated. The simulation model consists of differential drive mobile robot equipped with rotating ultrasonic system. Control algorithm is developed and implemented to control the rotation behavior of the system and to extract ultrasonic sensor readings. The previous simulation arrangements are repeated with differential drive mobile robot equipped with three fixed ultrasonic sensors. Ultrasonic sensors readings of both simulation arrangements are recorded, displayed and discussed.

Simulation Tools

In this research, Gazebo simulator is used to simulate the physical structure and dynamic characteristics of differential drive mobile robot, servomotor and ultrasonic sensor. Gazebo simulator project is built by C++, it integrates Open Dynamic Engine library for rigid bodies dynamic simulation [11] and OpenGL library for rendering and visualization. ROS (Robotic Operating System) is used to control the simulated items in Gazebo, and to extract simulation data from Gazebo environment. ROS is a combination of libraries and tools built by C++ and Python especially to develop and build robot applications [12]. Python language is used to process the extracted results, numpy library [13] is used for data filtering and analysis and matplotlib [14] library for chart plotting and formatting.

Simulation Parameters



(a) Rotating ultrasonic sensor model.

(b) Fixed ultrasonic sensors model.

FIGURE 2. Simulation environment for rotating and fixed range sensor approaches.

Two ultrasonic sensors approaches are modeled. The first model is used to simulate the rotating ultrasonic sensor approach and it consists of differential drive equipped with one rotating ultrasonic sensor in the middle front of the vehicle. The second model is used to simulate the fixed ultrasonic sensor approach and it consists of differential drive equipped with three fixed ultrasonic sensors distributed in the middle front middle, left and right of the vehicle.

In the first model, the simulated ultrasonic sensor has the following parameters: $\theta_{range} = 20^\circ$, $d_{range} = 1\,m$ and the desired covered area has $a_{total} = \frac{\pi}{2} d_{range}^2$, $\theta_{start} = 0^\circ$ and $\theta_{end} = 180^\circ$. From equation (1), (2) we get the total number of stops $N = 9\,stops$ and the values of each θ_{stop} as following:

$$\theta_{stop} = \{10^\circ, 30^\circ, 50^\circ, 70^\circ, 90^\circ, 110^\circ, 130^\circ, 150^\circ, 170^\circ\}$$

To calculate the total rotating time t_{total} for one cycle over a_{total}, we have to calculate t_{move} and t_{stop}. To get the t_{move}, we get the ultrasonic rotation angular velocity, which is the same as servomotor angular velocity ω_m. In practical experiments ω_m is described in servomotor data sheet, in this simulation we use the value $\omega_m = 100\,deg/sec$ and $t_{move} = 0.2\,sec$. Stop time t_{stop} is the ultrasonic maximum time needed to measure the distance to an obstacle inside sensor's range d_{range}. This can be obtained by calculating the time that ultrasonic wave takes to propagate forward and backward for a distance d_{range} , knowing that speed of ultrasonic wave is approximately $343\,m/sec$ at ambient temperature 20° we get:

$$t_{stop} = t_{trig} + \frac{2d_{range}}{343} \approx 5.8\,msec$$

Trigger time t_{trig} is the time needed by ultrasonic sensor to trigger the emitter to emit the ultrasonic wave. t_{trig} is a very small amount of time near $10\,\mu sec$ and we neglect this amount in our calculations assuming $t_{trig} = 0\,sec$. Now, t_{total} is calculated from equation (3) to get $t_{total} = 1.6522\,sec$.

In the second model, three ultrasonic sensors with the same parameters as the one used in the first model are fixed on the vehicle chassis. Sensors heading angles are $\theta_r = 10^\circ$, $\theta_c = 90^\circ$ and $\theta_l = 170^\circ$. Both models are driven with the same linear velocity to pass by an obstacle represented by a box located at the left side model path.

Rotating Ultrasonic Controller Algorithm

In order to control the rotation behavior of the ultrasonic sensor in Gazebo simulator, a controller algorithm is programmed by C++ and implemented inside a ROS node. The main functions of this controller node is to control differential drive linear velocity, control ultrasonic angular velocity, stop times and stop angles. It also extracts the following data; simulation time, differential drive position, sensor position and sensor range readings, then it saves them as CSV (comma separated values) to be analyzed by Python. Controller algorithm uses ros-control package [15] to control the rotation of ultrasonic sensor and the differential drive actuators. In figure 3, the path of data transfer between simulation nodes is shown by using rqt-graph package in ROS. The key part is the controller algorithm which is implemented in (/control) node. this node publishes the angular velocity and θ_{stop} values to the ultrasonic sensor inside gazebo simulator over the topic (/ddmr_joint_sensor/command). Controller node also gets the odometry message over (/my_odom) topic from the publisher node (/ddmr/odom_pub) that includes data about differential drive position and ultrasonic sensor range readings.

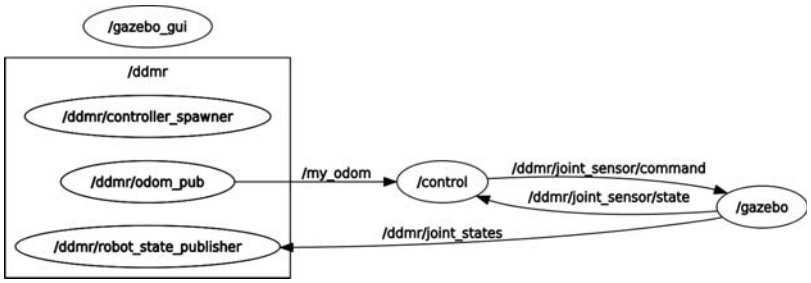


FIGURE 3. ROS messages and topics directions of controller algorithm node.

Control algorithm flowchart is shown in figure 4 and it starts by setting the initial state of ultrasonic angle $\theta_{stop}(1)$ and direction of rotation is counterclockwise. Then, control algorithm checks if the ultrasonic sensor rotates a full cycle or not by comparing the number of stops done by the sensor n with the total numbers of stops N calculated from equation (1). If n less than N, this means that sensor still hasn't scanned a complete cycle and it continue to hold for one more t_{stop} to collect range data. Then, controller record the information about value of current stop theta $\theta_{stop}(n)$, sensor range reading $d_{reading}$ and the current position of the differential drive $ddmr_{odom}$. The equation to calculate the next stop angle $\theta_{stop}(n-1)$ is determined according to the rotating direction counterclockwise (ccw) or clockwise (cw). Then, the controller publishes command to move the sensor to the next stop angle and updates the value of current stop angle. Finally, the controller loops to repeat the previous steps till n equals N which means that rotating sensor has finished to scan a complete cycle and the controller resets the stop angle counter value to $n = 1$ and switch the rotating direction.

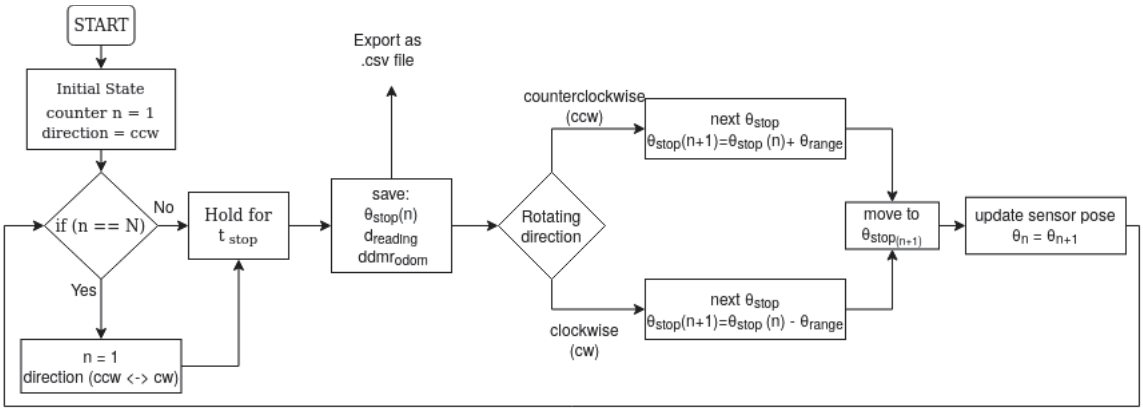
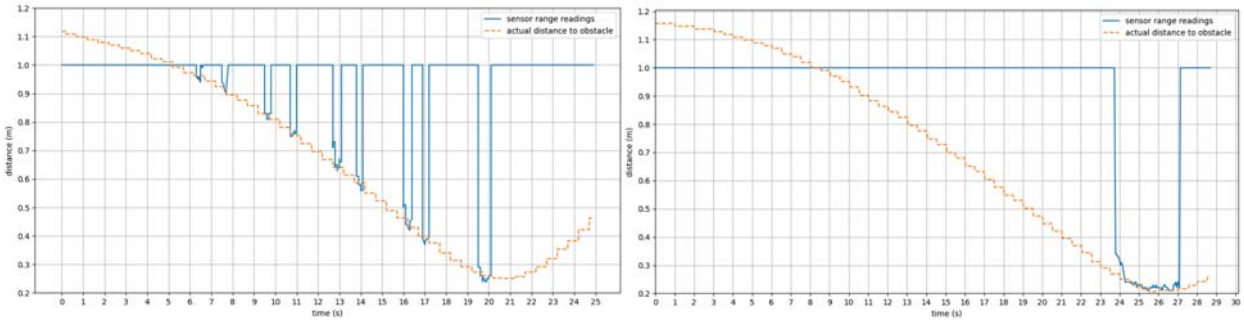


FIGURE 4. Control algorithm flowchart of the rotating range sensor model.

RESULTS

At the beginning of simulation, each fixed and rotating sensor approaches are mounted on separate differential drives. Each differential drive initial position at the origin point ($x = 0$, $y = 0$) and moves with linear velocity along the x-axis as shown in figure 2. One obstacle is represented as a box with $0.1m$ dimension located at ($x = 1.2$, $y = 0.3$). Results of sensor readings is shown in figure 5. In the rotating sensor model, there is only one sensor. In the fixed sensors approach, there are three sensors, here the results show only the left sensor readings as it is heading to the direction of the obstacle. The other two sensors will give always d_{range} as there is no obstacle in their heading direction along the differential drive path. Results show two lines, the continuous lines represent the sensors distance readings in each model, and the dashed lines represents the actual distance to the obstacle.



(a) Rotating ultrasonic sensor model.

(b) Fixed ultrasonic sensor model.

FIGURE 5. Simulation result: ultrasonic sensor distance readings.

The rotating approach succeeded to detect the obstacle nine times from the moment that the actual distance to the obstacle is less than sensor range d_{range}. On the other hand, fixed approach detected the obstacle only one time. Blind area is the area on figure 5 where the distance reading by range sensor (continuous line) is higher than the actual distance value to the obstacle (dashed line). Fixed sensor model has one blind continuous area looks like a triangle. However in rotating model there are ten trapezoid blind areas. This is one of the key advantages of rotating sensor approach that they don't have continuous blind area, and each portion inside the covered area a_{area} is being periodically scanned by the rotating sensor.

CONCLUSION

Rotating range sensor approach is efficient to be used in mobile robot application for navigation and collision avoidance purposes. As shown in the Results section that rotating sensor approach has discontinuous blind area, the matter that gives it advantage over conventional fixed range sensors that equipped in more than one location on vehicle chassis. Rotating range sensor approach has a limitation depends on the mobile critical time $t_{critical}$. This limitation can be eliminated by improving the rotating approach, this can be done by dividing the total covered area a_{total} into smaller areas and integrate more than one rotating range sensor for each area. In addition, rotating range sensor approach can be an efficient backup for applications that use fixed range sensor system, In case of any damage happened to one or more of the fixed sensors during operation.

REFERENCES

1. F. Sweeney, M. Beckerman, P. Butler, J. Jones, and D. Reister, "Application of autonomous robotics to surveillance of waste storage containers for radioactive surface contamination," Tech. Rep. (Oak Ridge National Lab., 1991).
2. H. Everett, *Sensors for mobile robots* (CRC Press, 1995).
3. V. Zhmud, N. Kondratiev, K. Kuznetsov, V. Trubin, and L. Dimitrov, "Application of ultrasonic sensor for measuring distances in robotics," in *Journal of Physics: Conference Series*, Vol. 1015 (IOP Publishing, 2018) p. 032189.
4. R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots* (MIT press, 2011).

5. S. Yuta, S. Suzuki, and S. Iida, "Implementation of a small size experimental self-contained autonomous robot—sensors, vehicle control, and description of sensor based behavior," in *2nd International Symposium on Experimental Robotics, ISER 1991* (Springer Verlag, 1993) pp. 344–358.
6. K. Kimoto *et al.*, "Autonomous mobile robot simulator—a programming tool for sensor-based behavior," Autonomous Robots **1**, 131–148 (1995).
7. M. Mazo, F. J. Rodriguez, J. L. Lazaro, J. Urena, J. C. Garcia, E. Santiso, P. Revenga, and J. J. Garcia, "Wheelchair for physically disabled people with voice, ultrasonic and infrared sensor control," Autonomous Robots **2**, 203–224 (1995).
8. A. Budianto, R. Pangabidin, M. Syai'in, R. Adhitya, L. Subiyanto, A. Khumaidi, I. Rachman, B. Widiawan, K. Joni, E. Nurcahya, *et al.*, "Analysis of artificial intelligence application using back propagation neural network and fuzzy logic controller on wall-following autonomous mobile robot," in *2017 International Symposium on Electronics and Smart Devices (ISESD)* (IEEE, 2017) pp. 62–66.
9. R. Wang, L. Chen, J. Wang, P. Zhang, Q. Tan, and D. Pan, "Research on autonomous navigation of mobile robot based on multi ultrasonic sensor fusion," in *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)* (IEEE, 2018) pp. 720–725.
10. I. Ahmed and M. Y. Filimonov, "Collision avoidance algorithm with performance optimization and speed control for multi-robot autonomous system," in *48th International Youth School Conference "Modern Problems in Mathematics and its Applications", SoProMat 2017*, CEUR-WS Proceedings, Vol. 1894 (2017) pp. 115–122.
11. R. Smith, "Open dynamic engine user guide, 2006," URL http://www. ode. org, Last visited at **7** (2013).
12. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, Vol. 3 (Kobe, Japan, 2009) p. 5.
13. T. E. Oliphant, *A guide to NumPy*, Vol. 1 (Trelgol Publishing USA, 2006).
14. J. Hunter and D. Dale, "The matplotlib user's guide," Matplotlib 0.90. 0 user's guide (2007).
15. S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ros_control: A generic and simple control framework for ros," The Journal of Open Source Software (2017), 10.21105/joss.00456.